

A Truthful Randomized Mechanism for Combinatorial Public Projects via Convex Optimization*

Shaddin Dughmi[†]
 Department of Computer Science
 Stanford University
 shaddin@cs.stanford.edu

January 13, 2013

Abstract

In *Combinatorial Public Projects*, there is a set of projects that may be undertaken, and a set of self-interested players with a stake in the set of projects chosen. A public planner must choose a subset of these projects, subject to a resource constraint, with the goal of maximizing social welfare. Combinatorial Public Projects has emerged as one of the paradigmatic problems in *Algorithmic Mechanism Design*, a field concerned with solving fundamental resource allocation problems in the presence of both selfish behavior and the computational constraint of polynomial-time.

We design a polynomial-time, truthful-in-expectation, $(1 - 1/e)$ -approximation mechanism for welfare maximization in a fundamental variant of combinatorial public projects. Our results apply to combinatorial public projects when players have valuations that are *matroid rank sums (MRS)*, which encompass most concrete examples of submodular functions studied in this context, including coverage functions, matroid weighted-rank functions, and convex combinations thereof. Our approximation factor is the best possible, assuming $P \neq NP$. Ours is the first mechanism that achieves a constant factor approximation for a natural NP-hard variant of combinatorial public projects.

*Extended abstract appears in *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, 2011.

[†]Supported by NSF Grant CCF-0448664.

1 Introduction

The overarching goal of *algorithmic mechanism design* is to design computationally efficient algorithms that solve or approximate fundamental resource allocation problems in which the underlying data is a priori unknown to the algorithm. A problem that has received much attention in this context — albeit mostly in the form of negative results — is *Combinatorial Public Projects* (CPP). Here, there are m projects being considered by a public planner, n players, and a bound $k \leq m$ on the number of projects that may be chosen. Each player i has a private valuation $v_i(S)$ for each subset S of the projects. We consider the *flexible* variant of CPP, where a feasible solution is a set of at most k projects¹. The goal is to choose a feasible set of projects S maximizing *social welfare*: $\sum_i v_i(S)$. The valuations are initially unknown to the public planner, and must be elicited from the (self-interested) players. A “mechanism” for CPP extracts this information, and decides on a set of projects to undertake. The mechanisms we consider can charge the players payments in order to incentivize truthful reporting of their valuations. Moreover, we seek mechanisms that run in polynomial time.

Since CPP is highly inapproximable for general valuations — even by non-truthful algorithms — it is most interesting to study CPP for restricted classes of valuations. Most notable among these are submodular valuations, as they naturally model the pervasive notion of “diminishing marginal returns”. In this paper, we study CPP for a fundamental and large subset of submodular valuations: *Matroid Rank Sum Valuations*. This class includes most concrete examples of submodular functions studied in this context. Most notably, it includes the canonical and arguably most natural example of submodularity: coverage functions.

Combinatorial public projects and its variants are examples of *welfare maximization problems*. There are many other examples, most notable among them are *combinatorial auctions*, with their many variants (see e.g. [25]). Welfare maximization problems occupy a central position in mechanism design, not only because of the fundamental nature of the utilitarian objective, but also due to the rich economic theory surrounding them. Most notably, the celebrated Vickrey-Clarke-Groves (VCG) mechanism (see e.g. [25]) is a general solution for all these problems, at least from an economic perspective. The VCG mechanism is *truthful*, in that it is in a player’s best interest to report his true valuations regardless of the reports of the other players. Moreover, VCG finds the welfare maximizing solution.

Unfortunately, however, most interesting welfare maximization problems, such as combinatorial public projects, are NP-hard. Therefore, implementing VCG efficiently — i.e. in polynomial time — is impossible unless $P = NP$. Moreover, as first argued in [24], most existing approximation algorithms — unlike exact algorithms — cannot be converted to truthful mechanisms by the imposition of a suitable payment scheme. This necessitates the design of carefully crafted approximation algorithms, tailored specifically for truthfulness. Understanding the power of these truthful approximation mechanisms is the central goal of algorithmic mechanism design. This research agenda was first advocated by Nisan and Ronen [23]. Since then, combinatorial auctions and combinatorial public projects have emerged as the paradigmatic “challenge-problems” of the field, with much work in recent years establishing upper and lower-bounds on truthful polynomial-time mechanisms for these problems, for example: [20, 11, 13, 12, 10, 6, 14, 27, 3, 4, 7, 17].

The “holy grail” of algorithmic mechanism design is to design polynomial-time truthful approximation mechanisms that match the approximation guarantee of the best (non-truthful) polynomial-time approximation algorithm. Unfortunately, several recent impossibility results have shed serious doubt on the possibility

¹This is in contrast to the *exact* variant, where each feasible solution consists of *exactly* k projects — a difference that is uninteresting in an approximation algorithms context, yet has major implications when incentives are in the picture. For more on the distinction between the two variants, we refer the reader to [7].

of this goal [10, 27, 3, 4, 7]. Combinatorial public projects, in particular, bore the brunt of the most brutal of these negative results [27, 4, 7]. Fortunately, all but one of these lower bounds apply exclusively to deterministic mechanisms, and none apply to randomized mechanisms for the — arguably more natural — flexible variant of combinatorial public projects.

As the limitations of deterministic mechanisms became apparent, a recent research direction has focused on designing randomized approximation mechanisms for the fundamental problems of algorithmic mechanism design [20, 8, 15, 9, 17]. These mechanisms are instances of the only general approach² known for designing (randomized) truthful mechanisms: via *maximal-in-distributional range (MIDR) algorithms* [8]. An MIDR algorithm fixes a set of distributions over feasible solutions — the *distributional range* — independently of the valuations reported by the self-interested participants, and outputs a random sample from the distribution that maximizes expected (reported) welfare. The “Vickrey-Clarke-Groves (VCG)” payment scheme renders an MIDR algorithm *truthful-in-expectation* — that is, a player unaware of the coin flips of the mechanism maximizes his expected utility by reporting truthfully.

Recently Dughmi, Roughgarden and Yan [17] presented the most general framework to date for the design of maximal-in-distributional-range algorithms. Their approach is based on convex optimization, and generalizes the celebrated linear-programming based approach of Lavi and Swamy [20]. Given a mathematical relaxation to a welfare maximization problem, [17] advocates designing randomized rounding schemes that are *convex*. Given a convex rounding scheme, the problem of finding the best *output* of the rounding scheme is a convex optimization problem solvable in polynomial time, and implements an MIDR allocation rule. They then show how to design a convex rounding scheme for combinatorial auctions with matroid rank sum valuations, yielding an optimal $(1 - 1/e)$ approximation mechanism. We elaborate on the framework of [17] in Section 2.5.

By reducing the problem of designing a truthful mechanism to that of designing a convex rounding scheme, the approach of [17] yielded the first optimal truthful mechanism for a variant of combinatorial auctions with restricted valuations. It is now natural to wonder if their approach is applicable to other welfare maximization problems. In particular, can the convex rounding framework be used to obtain optimal approximation mechanisms for interesting variants of Combinatorial Public Projects?

We answer this question in the affirmative, and elaborate on our contributions below.

1.1 Contributions

We design a $(1 - 1/e)$ -approximate convex rounding scheme for combinatorial public projects with matroid rank sum valuations. This yields a $(1 - 1/e)$ -approximate truthful-in-expectation mechanism for CPP, running in expected polynomial-time. This is the best approximation possible for this problem, even without truthfulness, unless $P = NP$. Therefore, ours is the first truthful mechanism for an NP-hard variant of CPP that matches the approximation ratio of the best non-truthful algorithm. Our results works with “black-box” valuations, provided that players can answer a randomized analogue of value oracles.

To prove our results, we follow the general outline of [17]. However, our task is more challenging: whereas in combinatorial auctions, randomized rounding may allocate each item independently (the approach taken in [17]), this is not possible in CPP. We must respect the cardinality constraint of k on the set of chosen projects, and therefore our rounding scheme must by fiat be *dependent*. This presents a major challenge in analyzing our rounding scheme. Whereas the expected value of a submodular function on a product distribution (i.e. independent rounding) has been studied extensively, and is closely related to the

²The random sampling approach used in [6], while arguably general, does not seem applicable beyond auction settings — in particular, it is not applicable to combinatorial public projects.

now well-understood multi-linear (see e.g. [5, 30]), analyzing the expected value of a dependent distribution — in particular proving it to be a concave function of underlying parameters — is a technical challenge that we overcome by combining techniques from combinatorics, convex analysis, and matroid theory.

1.2 Additional Related Work

Combinatorial Public Projects, in particular its *exact* variant, was first introduced by Papadimitriou, Schapira and Singer [27]. They show that no deterministic truthful mechanism for exact CPP with submodular valuations can guarantee better than a $O(\sqrt{m})$ approximation to the optimal social welfare. The non-strategic version of the problem, on the other hand, is equivalent to maximizing a submodular function subject to a cardinality constraint, and admits a $(1 - 1/e)$ -approximation algorithm due to Nemhauser, Wolsey and Fisher [21], and this is optimal [28] assuming $P \neq NP$.

Buchfuhrer, Schapira and Singer [4] explored approximation algorithms and truthful mechanisms for CPP with various classes of valuations in the submodular hierarchy. The most relevant result of [4] to our paper is a lower-bound of $O(\sqrt{m})$ on *deterministic* truthful mechanisms for the exact variant of CPP with coverage valuations — a class of valuations for which our *randomized* mechanism for flexible CPP obtains a $(1 - 1/e)$ approximation.

Most recently, Dobzinski [7] showed two lower bounds for CPP in the value oracle model: A lower bound of $O(\sqrt{m})$ on universally truthful mechanisms for flexible CPP with submodular valuations, and a lower bound of $O(\sqrt{m})$ on truthful-in-expectation mechanisms for *exact* CPP with submodular valuations. We note that the latter was the first unconditional lower bound on truthful-in-expectation mechanisms.

2 Preliminaries

2.1 Combinatorial Public Projects

In *Combinatorial Public Projects* there is a set $[m] = \{1, \dots, m\}$ of *projects*, a cardinality bound k such that $0 \leq k \leq m$, and a set $[n] = \{1, \dots, n\}$ of *players*. Each player i has a valuation function $v_i : 2^{[m]} \rightarrow \mathbb{R}_+$ that is normalized ($v_i(\emptyset) = 0$) and monotone ($v_i(A) \leq v_i(B)$ whenever $A \subseteq B$). In this paper, we consider the *flexible* variant of combinatorial public projects: a feasible solution is a set $S \subseteq [m]$ of projects with $|S| \leq k$. Player i 's value for outcome S is equal to $v_i(S)$. The goal is to choose the feasible set S maximizing *social welfare*: $\sum_i v_i(S)$.

We consider Combinatorial Public Projects where each player's valuation v_i is known to lie in some set \mathcal{V} of valuation functions. We abbreviate the set of instances of CPP constrained to valuations \mathcal{V} as $\text{CPP}(\mathcal{V})$. As first defined in [27], CPP was considered with \mathcal{V} equal to the set of monotone submodular functions. In this paper, we focus on CPP with matroid-rank-sum (MRS) valuations — a large subset of monotone submodular functions.

2.2 Mechanism Design Basics

We consider direct-revelation mechanisms for combinatorial public projects. Fix m, n , and k , and let $\mathcal{S} = \{S \subseteq [m] : |S| \leq k\}$ denote the set of feasible solutions. A mechanism comprises an *allocation rule*, which is a function from (hopefully truthfully) reported valuation functions $v_1, \dots, v_n : 2^{[m]} \rightarrow \mathbb{R}$ to a feasible outcome $S \in \mathcal{S}$, and a *payment rule*, which is a function from reported valuation functions to a required payment from each player. We allow the allocation and payment rules to be randomized.

A mechanism with allocation and payment rules \mathcal{A} and p is *truthful-in-expectation* if every player always maximizes its expected payoff by truthfully reporting its valuation function, meaning that

$$\mathbf{E}[v_i(\mathcal{A}(v)) - p_i(v)] \geq \mathbf{E}[v_i(\mathcal{A}(v'_i, v_{-i})) - p_i(v'_i, v_{-i})] \quad (1)$$

for every player i , (true) valuation function v_i , (reported) valuation function v'_i , and (reported) valuation functions v_{-i} of the other players. The expectation in (1) is over the coin flips of the mechanism.

The mechanisms that we design can be thought of as randomized variations on the classical VCG mechanism, as we explain next. Recall that the *VCG mechanism* is defined by the (generally intractable) allocation rule that selects the welfare-maximizing outcome with respect to the reported valuation functions, and the payment rule that charges each player i a bid-independent “pivot term” minus the reported welfare earned by other players in the selected outcome. This (deterministic) mechanism is truthful; see e.g. [22].

Now let $\text{dist}(\mathcal{S})$ denote the probability distributions over the feasible set \mathcal{S} , and let $\mathcal{D} \subseteq \text{dist}(\mathcal{S})$ be a compact subset of them. The corresponding *Maximal-In-Distributional-Range (MIDR)* allocation rule is defined as follows: given reported valuation functions v_1, \dots, v_n , return an outcome that is sampled randomly from a distribution $D^* \in \mathcal{D}$ that maximizes the expected welfare $\mathbf{E}_{S \sim D}[\sum_i v_i(S)]$ over all distributions $D \in \mathcal{D}$. Analogous to the VCG mechanism, there is a (randomized) payment rule that can be coupled with this allocation rule to yield a truthful-in-expectation mechanism (see [8]).

2.3 Matroid Rank Sum Valuations

We now define matroid rank sum valuations. Relevant concepts from matroid theory are reviewed in Appendix B.1.

Definition 2.1. A set function $v : 2^{[m]} \rightarrow \mathbb{R}$ is a *matroid rank sum (MRS) function* if there exists a family of matroid rank functions $u_1, \dots, u_\kappa : 2^{[m]} \rightarrow \mathbb{R}$, and associated non-negative weights $w_1, \dots, w_\kappa \in \mathbb{R}^+$, such that $v(S) = \sum_{\ell=1}^\kappa w_\ell u_\ell(S)$ for all $S \subseteq [m]$.

We do not assume any particular representation of MRS functions, and require only oracle access to their (expected) values on certain distributions (see Section 2.4). MRS valuations include most concrete examples of monotone submodular functions that appear in the literature — this includes coverage functions³, matroid weighted-rank functions⁴, and all convex combinations thereof. Moreover, as shown in [28], $1 - 1/e$ is the best approximation possible for CPP with coverage valuations — and hence also for MRS valuations — in polynomial time, even ignoring strategic considerations. That being said, we note that some interesting submodular functions — such as some budget additive functions⁵ — are not in the matroid rank sum family.

2.4 Lotteries and Oracles

A *value oracle* for a valuation $v : 2^{[m]} \rightarrow \mathbb{R}$ takes as input a set $S \subseteq [m]$, and returns $v(S)$. We define an analogous oracle that takes in a description of a simple lottery over sets $S \subseteq [m]$, and outputs the expectation of v over this lottery.

³A coverage function f on ground set $[m]$ designates some set \mathcal{Y} , and m subsets $A_1, \dots, A_m \subseteq \mathcal{Y}$, such that $f(S) = |\cup_{\ell \in S} A_\ell|$. We note that \mathcal{Y} may be an infinite, yet measurable, space. Coverage functions are arguably *the* canonical example of a submodular function.

⁴This is a generalization of matroid rank functions, where weights are placed on elements of the matroid. It is true, though not immediately obvious, that a matroid weighted-rank function can be expressed as a weighted combination of matroid (unweighted) rank functions — see e.g. [16].

⁵A set function f on ground set $[m]$ is *budgeted additive* if there exists a constant $B \geq 0$ (the budget) such that $f(S) = \min(B, \sum_{j \in S} f(\{j\}))$.

Let $k \in [m]$, let $R \subseteq [m]$, and let $x \in [0, 1]^m$ be a vector such that $\sum_j x_j \leq 1$. We interpret x as a probability distribution over $[m] \cup \{*\}$, where $*$ represents not choosing a project. Specifically, project $j \in [m]$ is chosen with probability x_j , and $*$ is chosen with probability $1 - \sum_j x_j$. We define a distribution $D_k^R(x)$ over $2^{[m]}$, and call this distribution the *k-bounded lottery with marginals x and promise R* . We sample $S \sim D_k^R(x)$ as follows: Let j_1, \dots, j_k be independent draws from x , and let $S = R \cup \{j_1, \dots, j_k\} \setminus \{*\}$. Essentially, this lottery commits to choosing projects R , and adds an additional k projects chosen randomly with replacement from distribution x . When $R = \emptyset$, as will be the case through most of this paper, we omit mention of the promised set. We can now define a randomized analogue of a value oracle that returns the expected value of a bounded-lottery.

Definition 2.2. A bounded-lottery-value oracle for set function $v : 2^{[m]} \rightarrow \mathbb{R}$ takes as input a vector $x \in [0, 1]^m$ with $\sum_j x_j \leq 1$, a bound $k \in [m]$, and a set $R \subseteq [m]$, and outputs $\mathbf{E}_{S \sim D_k^R(x)}[v(S)]$.

In our model for CPP, we assume that a player with valuation function v_i can answer bounded-lottery-value oracle queries for v_i . A bounded-lottery-value oracle is a generalization of value oracles. Nevertheless, it is the case that a bounded-lottery-value oracle can be implemented using a value oracle for some succinctly represented examples of MRS valuations, such as explicit coverage functions (In similar fashion to [17, Appendix A]).

More generally we note that bounded-lottery-value oracles can be approximated arbitrarily well, with high probability, using value oracles; this is done by random sampling, and we omit the technical details. Unfortunately, we are not able to reconcile the incurred sampling errors — small as they may be — with the requirement that our mechanism be *exactly* truthful. We suspect that relaxing our solution concept to approximate truthfulness — also known as ϵ -truthfulness — would remove this difficulty, and allow us to relax our oracle model to the more traditional value oracles.

2.5 Convex Rounding

In this section, we review *convex rounding*, a framework for the design of truthful mechanisms introduced by Dughmi, Roughgarden and Yan [17]. We present the main definitions and lemmas as they pertain to combinatorial public projects. For a more thorough and general treatment of convex rounding, we refer the reader to [17, Section 3].

We consider the standard integer programming formulation of CPP. There is a variable $x_j \in \{0, 1\}$ for each project $j \in [m]$, and the goal is to set at most k of the variables to 1 so that the welfare $v(x) = \sum_i v_i(\{j : x_j = 1\})$ is maximized. We *relax* this integer program in the obvious way to the polytope $\mathcal{P} = \{x \in \mathbb{R}^m : \sum_j x_j \leq k, x \succeq 0\}$. We postulate a *rounding scheme* r that maps points of \mathcal{P} to the feasible solutions $\mathcal{S} = \{S \subseteq [m] : |S| \leq k\}$ of CPP. We allow r to be randomized, so that $r(x)$ is a distribution over \mathcal{S} for each $x \in \mathcal{P}$.

Traditionally, approximation algorithms optimize an objective $\tilde{v}(x)$ — often a simple extension of v to \mathcal{P} — over the set \mathcal{P} of fractional solutions, and then round the optimal fractional point x^* to a solution $r(x^*)$ in the original feasible set \mathcal{S} . Many of the best approximation algorithms for various problems are based on this relax-solve-round framework. Unfortunately, however, this approach is almost always incompatible with the design of truthful mechanisms, due to the fact that the rounding step is often unpredictable. Truthful mechanism design, on the other hand, is intimately tied to *exact optimization*, as evidenced by the fact that the vast majority truthful mechanisms for multi-parameter problems are based on the VCG paradigm (see Section 2.2).

In an effort to reconcile the techniques of approximation algorithms and truthful mechanism design, Dughmi, Roughgarden and Yan proposed *optimizing directly on the output of the rounding scheme, rather than on its input*. This defines an optimization problem induced by relaxation \mathcal{P} and rounding scheme r . Stated for CPP with the relaxation as described above, the problem is as follows.

$$\begin{aligned} & \text{maximize} && \mathbf{E}_{S \sim r(x)} [\sum_i v_i(S)] \\ & \text{subject to} && \sum_{j=1}^m x_j \leq k \\ & && 0 \leq x_j \leq 1, \quad \text{for } j = 1, \dots, m. \end{aligned} \tag{2}$$

They consider a simple allocation rule, which we state for CPP in Algorithm 1, that solves (2) optimally. They observe that this allocation rule is maximal-in-distributional-range.

Algorithm 1 MIDR Allocation Rule for CPP

Parameter: n, m, k

Parameter: (Randomized) rounding scheme r

Input: Valuation functions $\{v_i\}_{i=1}^n$

Output: A set $S \subseteq [m]$ with $|S| \leq k$

- 1: Let x^* be an optimal solution to (2)
 - 2: Let $S \sim r(x^*)$
-

Lemma 2.3 ([17]). *Algorithm 1 is an MIDR allocation rule.*

For $\alpha \leq 1$, we say that the rounding scheme r for $\text{CPP}(\mathcal{V})$ is α -approximate if, whenever x is an integer point of \mathcal{P} corresponding to a set $S \in \mathcal{S}$, and $v_i \in \mathcal{V}$ for each i , we have that $\mathbf{E}_{T \sim r(x)} [\sum_i v_i(T)] \geq \alpha \sum_i v_i(S)$. In other words, rounding does not degrade the quality of an integer solution by more than α . Given the definition of Algorithm 1, it is easy to conclude the following lemma.

Lemma 2.4 ([17]). *If r is an α -approximate rounding scheme for $\text{CPP}(\mathcal{V})$, then Algorithm 1 is an α -approximation algorithm for $\text{CPP}(\mathcal{V})$.*

For reasons outlined in [17], implementing Algorithm 1 efficiently is impossible for most rounding schemes r in the literature. To get around this difficulty, they advocate designing rounding schemes that render (2) a convex optimization problem.

Definition 2.5. *Consider a randomized rounding scheme $r : \mathcal{P} \rightarrow \text{dist}(\mathcal{S})$. We say r is a convex rounding scheme for $\text{CPP}(\mathcal{V})$ if, whenever $v_i \in \mathcal{V}$ for all i , the objective $\mathbf{E}_{S \sim r(x)} [\sum_i v_i(S)]$ is a concave function of x .*

Lemma 2.6. *When r is a convex rounding scheme for $\text{CPP}(\mathcal{V})$, (2) is a convex optimization problem for each instance of $\text{CPP}(\mathcal{V})$.*

Under additional technical conditions, discussed in the context of combinatorial public projects in Appendix A, convex program (2) can be solved efficiently (e.g., using the ellipsoid method). This reduces the design of a polynomial-time α -approximate MIDR algorithm to designing a polynomial-time α -approximate convex rounding scheme.

Summarizing, Lemmas 2.3, 2.4, and 2.6 give the following informal theorem.

Theorem 2.7 (Informal). *If there exists an α -approximate convex rounding scheme for $\text{CPP}(\mathcal{V})$, then there exists a truthful-in-expectation, polynomial-time, α -approximate mechanism for $\text{CPP}(\mathcal{V})$.*

3 The Mechanism

In this section, we prove the main result.

Theorem 3.1. *There is a $(1 - 1/e)$ -approximate, truthful-in-expectation mechanism for combinatorial public projects with matroid rank sum valuations in the bounded-lottery-value oracle model, running in expected $\text{poly}(n, m)$ time.*

We structure the proof of Theorem 3.1 as follows. We define the k -bounded-lottery rounding scheme, which we denote by r_k , in Section 3.1. We prove that r_k is $(1 - 1/e)$ -approximate (Lemma 3.3), and convex (Lemma 3.2). Lemmas 2.3, 2.4 and 3.3, taken together, imply that Algorithm 1 when instantiated with $r = r_k$, is a $(1 - 1/e)$ -approximate MIDR allocation rule. Lemma 3.2 reduces implementing this allocation rule to solving a convex program.

In Appendix A, we handle the technical and numerical issues related to solving convex programs. First, we prove that our instantiation of Algorithm 1 can be implemented in expected polynomial-time using the ellipsoid method under a simplifying assumption on the numerical conditioning of our convex program (Lemma A.2). Then we show in Section A.3 that the previous assumption can be removed by slightly modifying our algorithm.

Finally, we prove that truth-telling VCG payments can be computed efficiently in Lemma B.4. Taken together, these lemmas complete the proof of Theorem 3.1.

3.1 The k -Bounded-Lottery Rounding Scheme

We devise a rounding scheme r_k that we term the k -bounded-lottery rounding scheme. Given a feasible solution x to linear program (2), we let distribution $r_k(x)$ be the k -bounded-lottery with marginals x/k (and promise \emptyset), as defined in Section 2.4. We make this more explicit in Algorithm 2.

Algorithm 2 The k -Bounded-Lottery Rounding Scheme r_k

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j .

Output: $S \subseteq [m]$ with $|S| \leq k$

- 1: For each $j \in [m]$ designate the interval $I_j = [\frac{1}{k} \sum_{j' < j} x_{j'}, \frac{1}{k} \sum_{j' \leq j} x_{j'}]$ of length $\frac{x_j}{k}$
 - 2: Draw p_1, \dots, p_k independently and uniformly from $[0, 1]$
 - 3: Let $S = \{j \in [m] : \{p_1, \dots, p_k\} \cap I_j \neq \emptyset\}$
-

The k -bounded-lottery rounding scheme is $(1 - 1/e)$ approximate and convex. We prove the approximation lemma below. As for convexity, we present a simplified proof for the special case of coverage valuations in Section 3.2, and present the proof for MRS valuations in Section 3.3.

Lemma 3.2. *The k -bounded-lottery rounding scheme is convex for CPP with MRS valuations.*

Lemma 3.3. *The k -bounded-lottery rounding scheme is $(1 - 1/e)$ -approximate when valuations are sub-modular.*

Proof. Fix n, m, k and $\{v_i\}_{i=1}^n$. Let $S \subseteq [m]$ be a feasible solution to CPP — i.e. $|S| \leq k$. Let 1_S be the vector with 1 in indices corresponding to S , and 0 otherwise. Let $T \sim r_k(1_S)$. We will first show that each element of $j \in S$ is included in T with probability at least $1 - 1/e$. Observe that T is the union of k independent draws from a distribution on $[m] \cup \{*\}$, where each time the probability of $j \in S$ is $1/k$. Therefore, the probability that j is included in T is $1 - (1 - 1/k)^k \geq 1 - 1/e$.

Submodularity now implies that $\mathbf{E}[v_i(T)] \geq (1 - 1/e) \cdot v_i(S)$ for each player i — this was proved in many contexts: see for example [19, Lemma 2.2], and the earlier related result in [18, Proposition 2.3]. This completes the proof. \square

3.2 Warmup: Convexity for Coverage Valuations

In this section, we prove a special case of Lemma 3.2 for coverage valuations. Recall that a coverage function f on ground set $[m]$ designates some set \mathcal{Y} , and m subsets $A_1, \dots, A_m \subseteq \mathcal{Y}$, such that $f(S) = |\cup_{j \in S} A_j|$.

Fix n, m, k and $\{v_i\}_{i=1}^n$. Assume that, for each player i , the valuation function $v_i : 2^{[m]} \rightarrow \mathbb{R}$ is a coverage function. We let $v(S) = \sum_i v_i(S)$ be the welfare of a solution S to CPP. It is an easy observation that the sum of coverage functions is also a coverage function. Therefore $v(S)$ is a coverage function. We let \mathcal{Y} be a set, and $A_1, \dots, A_m \subseteq \mathcal{Y}$, such that $v(S) = |\cup_{j \in S} A_j|$. While our proof extends easily to the case where \mathcal{Y} is an arbitrary measure space, we assume in this section that \mathcal{Y} is a finite set for simplicity.

Let \mathcal{P} denote the polytope of fractional solutions to CPP as given in (2). We now show that $\mathbf{E}_{S \sim r_k(x)}[v(S)]$ is a concave function of x for $x \in \mathcal{P}$, completing the proof of Lemma 3.2 for the special case of coverage valuations. Take an arbitrary $x \in \mathcal{P}$, and let $S \sim r_k(x)$ be a random variable. Using linearity of expectations, we can rewrite the expected welfare as follows.

$$\mathbf{E}[v(S)] = \mathbf{E}[|\cup_{j \in S} A_j|] = \sum_{\ell \in \mathcal{Y}} \Pr[\ell \in \cup_{j \in S} A_j]$$

Since the sum of concave functions is concave, showing that $\Pr[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{Y}$ suffices to complete the proof. For $\ell \in \mathcal{Y}$, let $T_\ell = \{j \in [m] : \ell \in A_j\}$ be the set of projects that “cover” ℓ . Let p_1, \dots, p_k and I_1, \dots, I_k be as in Algorithm 2. Note that $\{I_j\}_{j=1}^m$ are disjoint sub-intervals of $[0, 1]$, and $|I_j| = \frac{x_j}{k}$. We can rewrite the probability of covering ℓ as follows.

$$\begin{aligned} \Pr[\ell \in \cup_{j \in S} A_j] &= \Pr[S \cap T_\ell \neq \emptyset] \\ &= \Pr[\{p_1, \dots, p_k\} \cap \cup_{j \in T_\ell} I_j \neq \emptyset] \\ &= 1 - \Pr[\{p_1, \dots, p_k\} \cap \cup_{j \in T_\ell} I_j = \emptyset] \\ &= 1 - \prod_{t=1}^k \Pr[p_t \notin \cup_{j \in T_\ell} I_j] \\ &= 1 - \prod_{t=1}^k (1 - |\cup_{j \in T_\ell} I_j|) \\ &= 1 - \left(1 - \frac{\sum_{j \in T_\ell} x_j}{k}\right)^k. \end{aligned}$$

The final form is simply the composition of the concave function $g(y) = 1 - (1 - y/k)^k$ with the affine function $y \rightarrow \sum_{j \in T_\ell} x_j$. It is well known that composing a concave function with an affine function yields another concave function (see e.g. [2]). This completes the proof.

3.3 Convexity for Matroid Rank Sum Valuations

In this section, we will prove Lemma 3.2 in its full generality. First, we recall the *discrete hessian matrix*, as defined in [17].

Definition 3.4 ([17]). Let $v : 2^{[m]} \rightarrow \mathbb{R}$ be a set function. For $S \subseteq [m]$, we define the discrete Hessian matrix $\mathcal{H}_S^v \in \mathbb{R}^{m \times m}$ of v at S as follows:

$$\mathcal{H}_S^v(i, j) = v(S \cup \{i, j\}) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S) \quad (3)$$

for $i, j \in [m]$.

It was shown in [17] that the discrete hessian matrices are negative semi-definite for matroid rank sum functions.

Claim 3.5 ([17]). If $v : 2^{[m]} \rightarrow \mathbb{R}^+$ is a matroid rank sum function, then \mathcal{H}_S^v is negative semi-definite for each $S \subseteq [m]$.

We now return to Lemma 3.2. Fix n and m . For each cardinality bound $k \in [m]$, let \mathcal{P}_k denote the polytope of fractional solutions to CPP as given in (2). For a set of MRS valuations v_1, \dots, v_n , we observe that the social welfare $v(S) = \sum_{i=1}^n v_i(S)$ is — by the (obvious) fact that the sum of MRS functions is an MRS function — also an MRS function. Therefore, we will prove Lemma 3.2 by showing that, for each $k \in [m]$ and MRS function $v : 2^{[m]} \rightarrow \mathbb{R}$, the following function of $x \in \mathcal{P}_k$ is concave in x .

$$\begin{aligned} G_k^v(x) &= \mathbf{E}_{S \sim r_k(x)} [v(S)] \\ &= \sum_{S \subseteq [m]} v(S) \Pr[r_k(x) = S] \end{aligned} \quad (4)$$

We use techniques from combinatorics to write $\Pr[r_k(x) = S]$ in a form that will be easier to work with. For $T \subseteq [m]$, we use x_T as short-hand for $\sum_{j \in T} x_j$, and \bar{T} as short-hand for $[m] \setminus T$.

Claim 3.6. For each $k \in [m]$, $x \in \mathcal{P}_k$, and $S \subseteq [m]$

$$\Pr[r_k(x) = S] = -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\bar{R}}}{k}\right)^k \quad (5)$$

Proof. It is easy to see that $\Pr[r_k(x) = S]$ is equal to:

$$\Pr[r_k(x) \subseteq S] - \Pr\left[\bigvee_{j \in S} r_k(x) \subseteq S \setminus \{j\}\right] \quad (6)$$

Using the inclusion-exclusion principle, we can rewrite (6) as follows:

$$\Pr[r_k(x) \subseteq S] - \sum_{\emptyset \neq T \subseteq S} -1^{|T|-1} \Pr[r_k(x) \subseteq S \setminus T] \quad (7)$$

Letting $R = S \setminus T$ in (7), we get

$$\Pr[r_k(x) \subseteq S] - \sum_{R \subsetneq S} -1^{|S|-|R|-1} \Pr[r_k(x) \subseteq R] \quad (8)$$

We can easily simplify (8) to conclude that

$$\Pr[r_k(x) = S] = \sum_{R \subseteq S} -1^{|S|-|R|} \Pr[r_k(x) \subseteq R] \quad (9)$$

Next, we observe that the expression $\Pr[r_k(x) \subseteq R]$ can be expressed as a simple closed form in x . Let p_1, \dots, p_k and I_1, \dots, I_m be as in Algorithm 2. The event $r_k(x) \subseteq R$ occurs exactly when none of p_1, \dots, p_k land in the intervals corresponding to projects \overline{R} . Recalling that the interval I_j of project j has length x_j/k , we get that the probability of any particular p_t falling in $\cup_{j \in \overline{R}} I_j$ is exactly $x_{\overline{R}}/k$. Therefore, by the independence of the variables p_1, \dots, p_k , we get that

$$\Pr[r_k(x) \subseteq R] = \left(1 - \frac{x_{\overline{R}}}{k}\right)^k \quad (10)$$

Combining (9) and (10) completes the proof. \square

Building on Claim 3.6, we now express the Hessian matrix of G_k^v as a non-negative weighted sum of discrete Hessian matrices of v . We note that when $x \in \mathcal{P}_k$, it is easy to verify that $\frac{k-2}{k} \cdot x \in \mathcal{P}_{k-2}$, and therefore (11) is well-defined.

Claim 3.7. *For each $k \in [m]$, $x \in \mathcal{P}_k$, and $v : 2^{[m]} \rightarrow \mathbb{R}$, we have*

$$\nabla^2 G_k^v(x) = \frac{k-1}{k} \sum_{S \subseteq [m]} \Pr \left[r_{k-2} \left(\frac{k-2}{k} \cdot x \right) = S \right] \mathcal{H}_S^v \quad (11)$$

Proof. Fix $i, j \in [m]$, possibly with $i = j$. We work with G_k^v as defined in Equation (4), and plug in expression (5).

$$G_k^v(x) = \sum_{S \subseteq [m]} v(S) \cdot -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^k$$

Differentiating with respect to x_i and x_j gives:

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S \subseteq [m]} v(S) \cdot -1^{|S|} \sum_{R \subseteq S \setminus \{i, j\}} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2}$$

We group the terms by projecting S onto $[m] \setminus \{i, j\}$, and then we simplify the resulting expression.

$$\begin{aligned} \frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} &= \frac{k-1}{k} \sum_{S \subseteq [m] \setminus \{i, j\}} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} (v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i, j\})) \\ &= \frac{k-1}{k} \sum_{S \subseteq [m]} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} (v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i, j\})) \\ &= \frac{k-1}{k} \sum_{S \subseteq [m]} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \mathcal{H}_S^v(i, j) \end{aligned} \quad (12)$$

The second equality follows from the fact that $v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i, j\}) = 0$ when S includes either of i and j . The last equality follows by definition of \mathcal{H}_S^v .

Invoking Claim 3.6 with $k' = k - 2$ and $x' = \frac{k-2}{k} \cdot x$, and plugging the resulting expression into (12), we conclude that

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S \subseteq [m]} \Pr \left[r_{k-2} \left(\frac{k-2}{k} \cdot x \right) = S \right] \mathcal{H}_S^v(i, j).$$

\square

Claims (3.5) and (3.7) establish that, when v is MRS and $k \in [m]$, $\nabla^2 G_k^v(x)$ is a non-negative weighted sum of negative semi-definite matrices for each $x \in \mathcal{P}_k$. A non-negative weighted sum of negative semi-definite matrices is negative semi-definite. Therefore, the Hessian matrix of G_k^v is negative semi-definite at each $x \in \mathcal{P}_k$, and we conclude that G_k^v is a concave function on \mathcal{P}_k . This completes the proof of Lemma 3.2.

Acknowledgments

The author thanks Tim Roughgarden and Qiqi Yan for helpful discussions.

References

- [1] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.
- [2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [3] Dave Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos Papadimitriou, Michael Schapira, Yaron Singer, and Chris Umans. Inapproximability for VCG-based combinatorial auctions. In *Proc. 21st ACM Symp. on Discrete Algorithms (SODA)*, 2010.
- [4] David Buchfuhrer, Michael Schapira, and Yaron Singer. Computation and incentives in combinatorial public projects. In *Proc. 12th ACM Conf. on Electronic Commerce (EC)*, 2010.
- [5] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proc. 12th Intl. Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007.
- [6] Shahar Dobzinski. Two randomized mechanisms for combinatorial auctions. In *Proc. 10th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2007.
- [7] Shahar Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing (STOC)*, 2011.
- [8] Shahar Dobzinski and Shaddin Dughmi. On the power of randomization in algorithmic mechanism design. In *Proc. 50th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2009.
- [9] Shahar Dobzinski, Hu Fu, and Robert Kleinberg. Truthfulness via proxies. *CoRR*, abs/1011.3232, 2010.
- [10] Shahar Dobzinski and Noam Nisan. Limitations of VCG-based mechanisms. In *Proc. 38th ACM Symp. on Theory of Computing (STOC)*, 2007.
- [11] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proc. 36th ACM Symp. on Theory of Computing (STOC)*, 2005.
- [12] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proc. 37th ACM Symp. on Theory of Computing (STOC)*, 2006.

- [13] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proc. 17th ACM Symp. on Discrete Algorithms (SODA)*, 2006.
- [14] Shahar Dobzinski and Mukund Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In *Proc. 10th ACM Conf. on Electronic Commerce (EC)*, 2008.
- [15] Shaddin Dughmi and Tim Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *Proc. 51st IEEE Symp. on Foundations of Computer Science (FOCS)*, 2010.
- [16] Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue submodularity. In *Proc. 11th ACM Conf. on Electronic Commerce (EC)*, 2009.
- [17] Shaddin Dughmi, Tim Roughgarden, and Qiqi Yan. From convex optimization to randomized mechanisms: Toward optimal combinatorial auctions. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing (STOC)*, 2011.
- [18] Uriel Feige. On maximizing welfare where the utility functions are subadditive. In *Proc. 37th ACM Symp. on Theory of Computing (STOC)*, 2006.
- [19] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *Proc. 48th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2007.
- [20] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. 46th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2005.
- [21] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(3), 1978.
- [22] Noam Nisan. Introduction to mechanism design (for computer scientists). In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [23] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, 1999.
- [24] Noam Nisan and Amir Ronen. Computationally feasible VCG-based mechanisms. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC)*, 2000.
- [25] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.
- [26] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.
- [27] Christos Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *Proc. 49th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2008.
- [28] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcg characterization of np. In *Proc. 29th ACM Symp. on Theory of Computing (STOC)*, 1997.
- [29] Alexander Schrijver. *Combinatorial Optimization*. Springer, 2003.

- [30] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, 2008.

A Solving The Convex Program

In this section, we overcome some technical difficulties related to the solvability of convex programs. We follow the general outline of [17, Appendix B], modifying the proofs throughout in order to handle the additional technical difficulties specific to CPP. We show in Section A.1 that, in the bounded-lottery-value oracle model, the four conditions for “solvability” of convex programs, as stated in Fact B.3, are easily satisfied for convex program (2) when $r = r_k$. However, an additional challenge remains: “solving” a convex program — as in Definition B.2 — returns an approximately optimal solution. Indeed the optimal solution of a convex program may be irrational in general, so this is unavoidable.

We show how to overcome this difficulty if we settle for polynomial runtime in expectation. While the optimal solution x^* of (2) cannot be computed explicitly, the random variable $r_k(x^*)$ can be sampled in expected polynomial-time. The key idea is the following: *sampling the random variable $r_k(x^*)$ rarely requires precise knowledge of x^** . Depending on the coin flips of r_k , we decide how accurately we need to solve convex program (2) in order to compute $r_k(x^*)$. Roughly speaking, we show that the probability of requiring a $(1 - \epsilon)$ -approximation falls exponentially in $\frac{1}{\epsilon}$. As a result, we can sample $r_k(x^*)$ in expected polynomial-time. We implement this plan in Section A.2 under the simplifying assumption that convex program (2) is *well-conditioned* — i.e. is “sufficiently concave” everywhere. In Section A.3, we show how to remove that assumption by slightly modifying our algorithm.

A.1 Approximating the Convex Program

Claim A.1. *There is an algorithm for Combinatorial Public Projects with MRS valuations in the bounded-lottery-value oracle model that takes as input an instance of the problem and an approximation parameter $\epsilon > 0$, runs in $\text{poly}(n, m, \log(1/\epsilon))$ time, and returns a $(1 - \epsilon)$ -approximate solution to convex program (2) when $r = r_k$.*

It suffices to show that the four conditions of Fact B.3 are satisfied in our setting. The first three are immediate from elementary combinatorial optimization (see for example [29]). It remains to show that the first-order oracle, as defined in Fact B.3, can be implemented in polynomial-time in the bounded-lottery-value oracle model. We let $f(x)$ denote the objective function of convex program (2) when $r = r_k$. This objective can, by definition, be written as follows.

$$f(x) = \mathbf{E}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] = \sum_i G_k^{v_i}(x)$$

where v_i is the valuation function of player i and $G_k^{v_i}$ is as defined in (4). By definition, $G_k^{v_i}(x)$ is the outcome of querying the bounded-lottery-value oracle of v_i with bound k and marginals x/k . Therefore, we can evaluate $f(x)$ using n bounded-lottery-value queries, one for each player. It remains to show that we can also evaluate the (multi-variate) derivative $\nabla f(x)$ of $f(x)$. Using definition (4) and Claim 3.6, we take the partial derivative of $G_k^{v_i}$ with respect to x_j and simplify the resulting expression.

$$\begin{aligned}
\frac{\partial G_k^{v_i}}{\partial x_j}(x) &= \sum_{S \subseteq [m]} -1^{|S|} v_i(S) \sum_{R \subseteq S \setminus \{j\}} -1^{|R|+1} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m] \setminus \{j\}} -1^{|S|} (v_i(S \cup \{j\}) - v_i(S)) \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m]} -1^{|S|} (v_i(S \cup \{j\}) - v_i(S)) \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m]} v_i(S \cup \{j\}) \Pr \left[r_{k-1} \left(\frac{k-1}{k} x \right) = S \right] - \sum_{S \subseteq [m]} v_i(S) \Pr \left[r_{k-1} \left(\frac{k-1}{k} x \right) = S \right].
\end{aligned} \tag{13}$$

The second equality follows by grouping the terms of the summation by the projection of S onto $[m] \setminus \{j\}$. The third equality follows from the observation that $v(S \cup \{j\}) - v(S) = 0$ when S includes j . The fourth equality follows by a simple re-arrangement and application of Claim 3.6.

Inspect the final form (13) in light of the definition of bounded-lottery-value oracles (Definition 2.2) and the definition of r_k (Section 3.1). Notice that the first term is the expected value of v_i over the $(k-1)$ -bounded-lottery with marginals $\frac{k-1}{k}x$ and promise $\{j\}$. The second term is the expected value of v_i over the same lottery without the promise. Therefore, we can evaluate $\frac{\partial G_k^{v_i}}{\partial x_j}(x)$ using two queries to the bounded-lottery-value oracle of player i . This completes the proof of Claim A.1.

A.2 The Well-Conditioned Case

In this section, we make the following simplifying assumption: The objective function $f(x)$ of convex program (2) with $r = r_k$, when restricted to any line in the feasible set \mathcal{P} , has a second derivative of magnitude at least $\lambda = \frac{\sum_{i=1}^n v_i([m])}{2^{\text{poly}(n,m)}}$ everywhere, where the polynomial in the denominator may be arbitrary. This is equivalent to requiring that every eigenvalue of the Hessian matrix of $f(x)$ has magnitude at least λ when evaluated at any point in \mathcal{P} . Under this assumption, we prove Lemma A.2.

Lemma A.2. *Assume the magnitude of the second derivative of $f(x)$ is at least $\lambda = \frac{\sum_{i=1}^n v_i([m])}{2^{\text{poly}(n,m)}}$ everywhere. Algorithm 1, instantiated with $r = r_k$, can be simulated in time polynomial in n and m in expectation.*

Let x^* be the optimal solution to convex program (2) with $r = r_k$. Algorithm 1 outputs a set of projects distributed as $r_k(x^*)$. The k -bounded-lottery rounding scheme, as described in Algorithm 2, requires making k independent decisions: for $\ell \in \{1, \dots, k\}$, we draw p_ℓ uniformly from $[0, 1]$ and decide which interval I_j , if any, p_ℓ falls into. In other words, we find the minimum index j_ℓ (if any) such that $\sum_{j \leq j_\ell} x_j^*/k \geq p_\ell$. Fix ℓ . For most realizations of p_ℓ , we can calculate j_ℓ using only coarse estimates \tilde{x}_j to x_j^* . Assume we have an *estimation oracle* for x^* that, on input δ , returns a δ -estimate \tilde{x} of x^* : Specifically, $\tilde{x}_j - x_j^* \leq \delta$ for each $j \in [m]$. If p_ℓ falls outside the “uncertainty zones” of \tilde{x} , such as when $|p_\ell - \sum_{j' \leq j} \tilde{x}_{j'}/k| > \delta m/k$ for each $j \in [m]$, it is easy to see that we can correctly determine j_ℓ by using \tilde{x} in lieu of x . The total measure of the uncertainty zones of \tilde{x} is at most $2m^2\delta$, therefore p_ℓ lands outside the uncertainty zones with probability at least $1 - 2m^2\delta$. The following claim shows that if the estimation oracle for x^* can be implemented in time polynomial in $\log(1/\delta)$, then we can simulate the k -bounded-lottery rounding procedure in expected polynomial-time.

Claim A.3. Let x^* be the optimal solution of convex program (2) with $r = r_k$. Assume access to a subroutine $B(\delta)$ that returns a δ -estimate of x^* in $\text{poly}(n, m, \log(1/\delta))$ time. Algorithm 1, instantiated with $r = r_k$, can be simulated in expected $\text{poly}(n, m)$ time.

Proof. Fix $\ell \in \{1, \dots, k\}$. Draw $p_\ell \in [0, 1]$ uniformly at random as in the k -bounded-lottery rounding scheme in Algorithm 2. We will show how to find, in expected $\text{poly}(n, m)$ time, the minimum index j_ℓ (if any) such that $\sum_{j \leq j_\ell} x_j^*/k \geq p_\ell$.

The algorithm proceeds as follows: Start with $\delta = \delta_0 = \frac{1}{2m^2}$. Let $\tilde{x} = B(\delta)$. While $|p_\ell - \sum_{j' \leq j} \tilde{x}_{j'}/k| \leq \delta m/k$ for some $j \in [m]$ (i.e. p_ℓ may fall inside an “uncertainty zone”) do the following: let $\delta = \delta/2$, $\tilde{x} = B(\delta)$ and repeat. After the loop terminates, we have a sufficiently accurate estimate of x^* to calculate j_ℓ .

It is easy to see that the above procedure is a faithful simulation of Algorithm (2) on x^* . It remains to bound its expected running time. Let $\delta_t = \frac{1}{2^{t+1}m^2}$ denote the value of δ at iteration t . By our initial assumption, iteration t takes $\text{poly}(n, m, \log(1/\delta_t)) = \text{poly}(n, m, \log(2^{t+1}m^2)) = \text{poly}(n, m, t)$ time. The probability this procedure does not terminate after t iterations is at most $2m^2\delta_t = 1/2^t$. Taken together, these two facts and a simple geometric summation imply that the expected runtime is polynomial in n and m . \square

It remains to show that the estimation oracle $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time. At first blush, one may expect that the ellipsoid method can be used in the usual manner here. However, there is one complication: we require an estimate \tilde{x} that is close to x^* in *solution space* rather than in terms of objective value. Using our assumption on the curvature of $f(x)$, we will reduce finding a δ -estimate of x^* to finding an $1 - \epsilon(\delta)$ approximate solution to convex program (2) with $r = r_k$. The dependence of ϵ on δ will be such that $\epsilon \geq \text{poly}(\delta)/2^{\text{poly}(n, m)}$, thereby we can invoke Claim A.1 to deduce that $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time.

Let $\epsilon = \epsilon(\delta) = \frac{\delta^2 \lambda}{2 \sum_i v_i([m])}$. Plugging in the definition of λ , we deduce that $\epsilon \geq \delta^2/2^{\text{poly}(n, m)}$, which is the desired dependence. It remains to show that if \tilde{x} is $(1 - \epsilon)$ -approximate solution to (2), then \tilde{x} is also a δ -estimate of x^* .

Using the fact that $f(x)$ is concave, and moreover its second derivative has magnitude at least λ , it is a simple exercise to bound distance of any point x from the optimal point x^* in terms of its sub-optimality $f(x^*) - f(x)$, as follows:

$$f(x^*) - f(x) \geq \frac{\lambda}{2} \|x - x^*\|^2. \quad (14)$$

Assume \tilde{x} is a $(1 - \epsilon)$ -approximate solution to (2) with $r = r_k$. Equation (14) implies that

$$\|\tilde{x} - x^*\|^2 \leq \frac{2}{\lambda} \epsilon f(x^*) = \frac{\delta^2}{\sum_i v_i([m])} f(x^*) \leq \delta^2,$$

where the last inequality follows from the fact that $\sum_i v_i([m])$ is an upper-bound on the optimal value $f(x^*)$. Therefore, $\|x - x^*\| \leq \delta$, as needed. This completes the proof of Lemma A.2.

A.3 Guaranteeing Good Conditioning

In this section, we propose a modification r_k^+ of the k -bounded-lottery rounding scheme r_k . We will argue that r_k^+ satisfies all the properties of r_k established so far, with one exception: the approximation guarantee of Lemma 3.3 is reduced to $1 - 1/e - 2^{-2mn}$. Then we will show that r_k^+ satisfies the curvature assumption of Lemma A.2, demonstrating that said assumption may be removed. Therefore Algorithm 1, instantiated

with $r = r_k^+$ for combinatorial public projects with MRS valuations in the bounded-lottery-value oracle model, is $(1 - 1/e - 2^{-2mn})$ approximate and can be implemented in expected $\text{poly}(n, m)$ time. Finally, we show in Remark A.4 how to recover the 2^{-2mn} term to get a clean $1 - 1/e$ approximation ratio, as claimed in Theorem 3.1.

Let $\mu = 2^{-2nm}$. We define r_k^+ in Algorithm 3. Intuitively, r_k^+ first chooses a tentative set $S \subseteq [m]$ of projects using r_k . Then it cancels its choice with small probability μ . Finally, with probability β it chooses a random project $j^* \in [m]$ and lets $S = \{j^*\}$. β is defined as the fraction of projects included in the original tentative choice of S . The motivation behind this seemingly bizarre definition of r_k^+ is purely technical: as we will see, it can be thought of as adding “concave noise” to r_k .

Algorithm 3 Modified k -bounded-lottery Rounding Scheme r_k^+

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j .

Output: Feasible solution $S \subseteq [m]$ with $|S| \leq k$

```

1: Let  $S = r_k(x)$ 
2: Let  $\beta = \frac{|S|}{m}$ 
3: Draw  $q_1 \in [0, 1]$  uniformly
4: if  $q_1 \in [0, \mu]$  then
5:   Let  $S = \emptyset$ 
6:   Draw  $q_2 \in [0, 1]$  uniformly
7:   if  $q_2 \in [0, \beta]$  then
8:     Choose project  $j^* \in [m]$  uniformly at random.
9:     Let  $S = \{j^*\}$ 
10:  end if
11: end if

```

We can write the expected welfare $\mathbf{E}_{S \sim r_k^+(x)}[\sum_i v_i(S)]$ as follows.

$$\mathbf{E}_{S \sim r_k(x)} \left[(1 - \mu) \sum_i v_i(S) + \mu \beta \sum_i v_i(j^*) \right].$$

Using linearity of expectations and the fact that β is independent of the choice of j^* to simplify the expression, we get that $\mathbf{E}_{S \sim r_k^+(x)}[\sum_i v_i(S)]$ is equal to

$$(1 - \mu) \mathbf{E}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] + \mu \mathbf{E}[\beta] \frac{\sum_{j=1}^m \sum_{i=1}^n v_i(\{j\})}{m}.$$

Observe that r_k chooses a project j with probability $1 - (1 - x_j/k)^k$. Therefore, the expectation of β is $\frac{\sum_j 1 - (1 - x_j/k)^k}{m}$. This gives:

$$\mathbf{E}_{S \sim r_k^+(x)} \left[\sum_i v_i(S) \right] = (1 - \mu) \mathbf{E}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] + \frac{\mu}{m^2} \left(\sum_{j=1}^m \sum_{i=1}^n v_i(\{j\}) \right) \left(\sum_{j=1}^m 1 - (1 - x_j/k)^k \right). \quad (15)$$

It is clear that the expected welfare when using $r = r_k^+$ is within $1 - \mu = 1 - 2^{-2nm}$ of the expected welfare when using $r = r_k$ in the instantiation of Algorithm 1. Using Lemma 3.3, we conclude that r_k^+ is

a $(1 - 1/e - 2^{-2nm})$ -approximate rounding scheme. Moreover, using Lemma 3.2, as well as the fact that $1 - (1 - x_j/k)^k$ is a concave function, we conclude that r_k^+ is a convex rounding scheme. Therefore, this establishes the analogues of Lemmas 3.3 and 3.2 for r_k^+ . It is elementary to verify that our proof of Lemma A.2 can be adapted to r_k^+ as well.

It remains to show that r_k^+ is “sufficiently concave”. This would establish that the conditioning assumption of Section A.2 is unnecessary for r_k^+ . We will show that expression (15) is a concave function with curvature of magnitude at least $\lambda = \frac{\sum_{i=1}^n v_i([m])}{em^2 2^{2nm}}$ everywhere. Since the curvature of concave functions is always non-positive, and moreover the curvature of the sum of two functions is the sum of their curvatures, it suffices to show that the second term of the sum (15) has curvature of magnitude at least λ . We note that the curvature of $\sum_j (1 - (1 - x_j/k)^k)$ is at least e^{-1} over $x \in [0, 1]^m$. Therefore, the curvature of the second term of (15) is at least

$$\frac{\mu}{m^2} \left(\sum_i v_i([m]) \right) e^{-1} = \lambda$$

as needed.

Remark A.4. *In this section, we sacrificed 2^{-2nm} in the approximation ratio in order to guarantee expected polynomial runtime of our algorithm even when convex program (2) is not well-conditioned. This loss can be recovered to get a clean $1 - 1/e$ approximation as follows. Given our $(1 - 1/e - 2^{-2nm})$ -approximate MIDR algorithm \mathcal{A} , construct the following algorithm \mathcal{A}' : Given an instance of combinatorial public projects, \mathcal{A}' runs \mathcal{A} on the instance with probability $1 - e2^{-2nm}$, and with the remaining probability solves the instance optimally in exponential time $O(2^{2nm})$. It was shown in [15] that a random composition of MIDR mechanisms is MIDR, therefore \mathcal{A}' is MIDR. The expected runtime of \mathcal{A}' is bounded by the expected runtime of \mathcal{A} plus $e2^{-2nm} \cdot O(2^{2nm}) = O(1)$. Finally, the expected approximation of \mathcal{A}' is the weighted average of the approximation ratio of \mathcal{A} and the optimal approximation ratio 1, and is at least $(1 - e2^{-2nm})(1 - 1/e - 2^{-2nm}) + e2^{-2nm} \geq 1 - 1/e$.*

B Additional Preliminaries

B.1 Matroid Theory

In this section, we review some basics of matroid theory. For a more comprehensive reference, we refer the reader to [26].

A *matroid* M is a pair $(\mathcal{X}, \mathcal{I})$, where \mathcal{X} is a finite *ground set*, and \mathcal{I} is a non-empty family of subsets of \mathcal{X} satisfying the following two properties. (1) *Downward closure*: If S belongs to \mathcal{I} , then so do all subsets of S . (2) *The Exchange Property*: Whenever $T, S \in \mathcal{I}$ with $|T| < |S|$, there is some $x \in S \setminus T$ such that $T \cup \{x\} \in \mathcal{I}$. Elements of \mathcal{I} are often referred to as the *independent sets* of the matroid. Subsets of \mathcal{X} that are not in \mathcal{I} are often called *dependent*.

We associate with matroid M a set function $\text{rank}_M : 2^{\mathcal{X}} \rightarrow \mathbb{N}$, known as the *rank function* of M , defined as follows: $\text{rank}_M(A) = \max_{S \in \mathcal{I}} |S \cap A|$. Equivalently, the rank of set A in matroid M is the maximum size of an independent set contained in A . A set function f on a ground set \mathcal{X} is a *matroid rank function* if there exists a matroid M on the same ground set such that $f = \text{rank}_M$. Matroid rank functions are monotone ($f(S) \leq f(T)$ when $S \subseteq T$), normalized ($f(\emptyset) = 0$), and submodular ($f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all S and T).

B.2 Convex Optimization

In this section, we distill some basics of convex optimization. For more details, see [1].

Definition B.1. A maximization problem is given by a set Π of instances (\mathcal{P}, c) , where \mathcal{P} is a subset of some euclidean space, $c : \mathcal{P} \rightarrow \mathbb{R}$, and the goal is to maximize $c(x)$ over $x \in \mathcal{P}$. We say Π is a convex maximization problem if for every $(\mathcal{P}, c) \in \Pi$, \mathcal{P} is a compact convex set, and $c : \mathcal{P} \rightarrow \mathbb{R}$ is concave. If $c : \mathcal{P} \rightarrow \mathbb{R}^+$ for every instance of Π , we say Π is non-negative.

Definition B.2. We say a non-negative maximization problem Π is R -solvable in polynomial time if there is an algorithm that takes as input the representation of an instance $\mathcal{I} = (\mathcal{P}, c) \in \Pi$ — where we use $|\mathcal{I}|$ to denote the number of bits in the representation — and an approximation parameter ϵ , and in time $\text{poly}(|\mathcal{I}|, \log(1/\epsilon))$ outputs $x \in \mathcal{P}$ such that $c(x) \geq (1 - \epsilon) \max_{y \in \mathcal{P}} c(y)$.

Fact B.3. Consider a non-negative convex maximization problem Π . If the following are satisfied, then Π is R -solvable in polynomial time using the ellipsoid method. We let $\mathcal{I} = (\mathcal{P}, c)$ denote an instance of Π , and let m denote the dimension of the ambient euclidean space.

1. *Polynomial Dimension:* m is polynomial in $|\mathcal{I}|$.
2. *Starting ellipsoid:* There is an algorithm that computes, in time $\text{poly}(|\mathcal{I}|)$, a point $c \in \mathbb{R}^m$, a matrix $A \in \mathbb{R}^{m \times m}$, and a number $\mathcal{V} \in \mathbb{R}$ such that the following hold. We use $E(c, A)$ to denote the ellipsoid given by center c and linear transformation A .
 - (a) $E(c, A) \supseteq \mathcal{P}$
 - (b) $\mathcal{V} \leq \text{volume}(\mathcal{P})$
 - (c) $\frac{\text{volume}(E(c, A))}{\mathcal{V}} \leq 2^{\text{poly}(|\mathcal{I}|)}$
3. *Separation oracle for \mathcal{P} :* There is an algorithm that takes takes input \mathcal{I} and $x \in \mathbb{R}^m$, and in time $\text{poly}(|\mathcal{I}|, |x|)$ where $|x|$ denotes the size of the representation of x , outputs “yes” if $x \in \mathcal{P}$, otherwise outputs $h \in \mathbb{R}^m$ such that $h^T x < h^T y$ for every $y \in \mathcal{P}$.
4. *First order oracle for c :* There is an algorithm that takes input \mathcal{I} and $x \in \mathbb{R}^m$, and in time $\text{poly}(|\mathcal{I}|, |x|)$ outputs $c(x) \in \mathbb{R}$ and $\nabla c(x) \in \mathbb{R}^m$.

B.3 Computing Payments

Lemma B.4. Let \mathcal{A} be an MIDR allocation rule for combinatorial public projects, and let v_1, \dots, v_n be input valuations. Assume black-box access to \mathcal{A} , and value oracle access to $\{v_i\}_{i=1}^n$. We can compute, with $\text{poly}(n)$ over-head in runtime, payments p_1, \dots, p_n such that $\mathbf{E}[p_i]$ equals the VCG payment of player i for MIDR allocation rule \mathcal{A} on input v_1, \dots, v_n .

We note that an essentially identical lemma was proved in [17]. Nevertheless, we include a proof for completeness.

Proof. Without loss of generality, it suffices to show how to compute p_1 . Let $\mathbf{0} : 2^{[m]} \rightarrow \mathbb{R}$ be the valuation evaluating to 0 at each bundle. Recall (see e.g. [22]) that the VCG payment of player 1 is equal to

$$\mathbf{E}_{T \sim \mathcal{A}(\mathbf{0}, v_2, \dots, v_n)} \left[\sum_{i=2}^n v_i(T) \right] - \mathbf{E}_{S \sim \mathcal{A}(v_1, \dots, v_n)} \left[\sum_{i=2}^n v_i(S) \right]. \quad (16)$$

Let S be a sample from $\mathcal{A}(v_1, \dots, v_n)$, and let T be a sample from $\mathcal{A}(\mathbf{0}, v_2, \dots, v_n)$. Let $p_1 = \sum_{i=2}^n v_i(T) - \sum_{i=2}^n v_i(S)$. Using linearity of expectations, it is easy to see that the expectation of p_1 is equal to the expression in (16). This completes the proof. \square

We note that the mechanism resulting from Lemma B.4 is individually rational in expectation, and each payment is non-negative in expectation. We leave open the question of whether it is possible to enforce individual rationality and non-negative payments for our mechanism ex-post.